**What is Claimed:**

1. A video memory manager for a computer environment having a main processing unit for executing an operating system and an application, a system memory, and a graphics processing unit having a local video memory and an aperture that maps between a portion of the system memory and the graphics processing unit, the video memory manager comprising:

a physical memory manager that manages the physical memory of the local video memory and at least of portion of the physical memory of the system memory;

a graphics processing unit aperture manager that manages the memory mappings between a portion of system memory and the graphics processing unit, such that video data in the system memory is accessible to the graphics processing unit via the aperture; and

a virtual memory manager that allocates virtual memory and maintains mappings between the allocated virtual memory and the physical memory of the local video memory, the physical memory of the system memory, and the physical memory of the system memory via the aperture.

2. The video memory manager as recited in claim 1, wherein the physical memory manager further:

allocates memory for one of a driver resource and an application resource in the local video memory and does not evict the allocated memory from local video memory; and

the virtual memory manager further:

allocates and commits a kernel virtual address range for one of the driver resource and the application resource; and

maps the kernel virtual address range to the memory allocated in local video memory.

3. The video memory manager as recited in claim 1, wherein the virtual memory manager further:

allocates and commits a kernel virtual address range for one of a driver resource and an application resource; and

the physical memory manager further:

locks the committed kernel virtual address range, whereby the operating system does not have permission to page out one of the driver resource and the application resource from the system memory corresponding to the committed kernel virtual address range; and

the graphics processing unit aperture manager:

allocates an address range in the graphics processing unit aperture for redirection to one of the driver resource and the application resource; and

causes the allocated address range in the graphics processing unit aperture to be mapped to the committed kernel virtual address range.

4. The video memory manager as recited in claim 1, wherein the virtual memory manager further:

allocates and commits a kernel virtual address range for one of a driver resource and an application resource.

5. The video memory manager as recited in claim 4, wherein the physical memory manager further:

allocates memory in the local video memory for containing of one of the driver resource and the application resource; and

causes one of the driver resource and the application resource to be copied from memory corresponding to the committed kernel virtual address range to the memory allocated in the local video memory; and

the virtual memory manager further:

maps the kernel virtual address range to the memory allocated in the local video memory.

6. The video memory manager as recited in claim 5, wherein the physical memory manager further:

causes one of the driver resource and the application resource to be copied from the memory allocated in the local video memory to memory corresponding to the committed kernel virtual address range; and

frees the memory allocated in the local video memory; and

the virtual memory manager further:

frees the mapped kernel virtual address range.

7. The video memory manager as recited in claim 4, wherein the physical memory manager further:

locks the committed kernel virtual address range, whereby the operating system does not have permission to page out one of the driver resource and the application resource from the system memory corresponding to the committed kernel virtual address range; and

the graphics processing unit aperture manager further:

allocates an address range in the graphics processing unit aperture for redirection to one of the driver resource and the application resource; and

causes the address range in the graphics processing unit aperture to be mapped to the committed kernel virtual address range.

8. The video memory manager as recited in claim 7, wherein the graphics processing unit aperture manager further:

unmaps and frees the address range allocated in the graphics processing unit aperture; and

the physical memory manager further:

unlocks the committed kernel virtual address range, whereby the operating system has permission to page out one of the driver resource and the application resource from the system memory corresponding to the committed kernel virtual address range.

9. The video memory manager as recited in claim 1, wherein the virtual memory manager further:

allocates and commits an application private virtual address range for a surface, the surface not being directly accessible by the application via the main processing unit.

10. The video memory manager as recited in claim 9, wherein the physical memory manager further:

allocates memory in the local video memory for containing the surface; and

causes the surface to be copied from memory corresponding to the committed application private virtual address range to the memory allocated in the local video memory.

11. The video memory manager as recited in claim 10, wherein the physical memory manager further:

causes the surface to be copied from the memory allocated in the local video memory to memory corresponding to the committed application private virtual address range; and

frees the memory allocated in the local video memory.

12. The video memory manager as recited in claim 9, wherein the physical memory manager further:

locks the committed application private virtual address range, whereby the operating system does not have permission to page out the surface from the system memory corresponding to the application private virtual address range; and

the graphics processing unit aperture manager further:

allocates an address range in the graphics processing unit aperture for redirection to the surface; and

causes the address range in the graphics processing unit aperture to be mapped to the committed application private virtual address range.

13. The video memory manager as recited in claim 12, wherein the graphics processing unit aperture manager further:

unmaps and frees the allocated graphics processing unit aperture address range; and

the physical memory manager further:

unlocks the committed application private virtual address range, whereby the operating system has permission to page out the surface from the system memory corresponding to the committed application private virtual address range.

14. The video memory manager as recited in claim 1, wherein the virtual memory manager further:

allocates and commits an application private virtual address range for a surface, the surface being directly accessible by the application via the main processing unit.

15. The video memory manager as recited in claim 14, wherein the physical memory manager further:

allocates memory in the local video memory for containing the surface; and

causes the surface to be copied from the memory corresponding to the committed application private virtual address range to the memory allocated in the local video memory; and

the virtual memory manager further:

maps the committed application private virtual address range to the memory allocated in the local video memory.

16. The video memory manager as recited in claim 15, wherein the physical memory manager further:

causes the surface to be copied from the memory allocated in the local video memory to memory corresponding to the committed application private virtual address range; and

frees the allocated memory in the local video memory; and

the virtual memory manager further:

remaps the committed application private virtual address range to the memory corresponding to the committed application private virtual address range.

17. The video memory manager as recited in claim 14, wherein the physical memory manager further:

locks the committed application private virtual address range, whereby the operating system does not have permission to page out the surface from the system memory corresponding to the application private virtual address range; and

the graphics processing unit aperture manager further:

allocates an address range in the graphics processing unit aperture for redirection to the surface; and

causes the address range in the graphics processing unit aperture to be mapped to the committed application private virtual address range.

18. The video memory manager as recited in claim 17, wherein the graphics processing unit aperture manager further:

unmaps and frees the allocated graphics processing unit aperture address range; and

the physical memory manager further:

unlocks the committed application private virtual address range, whereby the operating system has permission to page out the surface from the system memory corresponding to the committed application private virtual address range.

19. A method for video memory management in a computer environment having a main processing unit for executing an operating system and an application, a system memory, and a graphics processing unit having a local video memory and an aperture that maps between a portion of system memory and the graphics processing unit, the method comprising:

managing the physical memory of the local video memory and at least of portion of the physical memory of the system memory;

managing the memory mappings between a portion of system memory and the graphics processing unit, such that video data in the system memory is accessible to the graphics processing unit via the aperture; and

allocating virtual memory and maintaining mappings between the allocated virtual memory and the physical memory of the local video memory, the physical memory of the system memory, and the physical memory of the system memory accessible via the aperture.

20. The method as recited in claim 19, further comprising:

allocating memory for one of a driver resource and an application resource in the local video memory and not evicting the allocated memory from local video memory;

allocating and committing a kernel virtual address range for one of the driver resource and the application resource; and

mapping the kernel virtual address range to the memory allocated in local video memory.

21. The method as recited in claim 19, further comprising:

allocating and committing a kernel virtual address range for one of a driver resource and an application resource;

locking the committed kernel virtual address range, whereby the operating system does not have permission to page out one of the driver resource and the application resource from the system memory corresponding to the committed kernel virtual address range;

allocating an address range in the graphics processing unit aperture for redirection to one of the driver resource and the application resource; and

causing the allocated address range in the graphics processing unit aperture to be mapped to the committed kernel virtual address range.

22. The method as recited in claim 19, further comprising:

allocating and committing a kernel virtual address range for one of a driver resource and an application resource.

23. The method as recited in claim 22, further comprising:

allocating memory in the local video memory for containing of one of the driver resource and the application resource;

causing one of the driver resource and the application resource to be copied from memory corresponding to the committed kernel virtual address range to the memory allocated in the local video memory; and

mapping the kernel virtual address range to the memory allocated in the local video memory.

24. The method as recited in claim 23, further comprising:

causing one of the driver resource and the application resource to be copied from the memory allocated in the local video memory to memory corresponding to the committed kernel virtual address range;

freeing the memory allocated in the local video memory; and

freeing the mapped kernel virtual address range.

25. The method as recited in claim 22, further comprising:

locking the committed kernel virtual address range, whereby the operating system does not have permission to page out one of the driver resource and the application resource from the system memory corresponding to the committed kernel virtual address range;

allocating an address range in the graphics processing unit aperture for redirection to one of the driver resource and the application resource; and

causing the address range in the graphics processing unit aperture to be mapped to the committed kernel virtual address range.

26. The method as recited in claim 25, further comprising:

unmapping and freeing the address range allocated in the graphics processing unit aperture; and

unlocking the committed kernel virtual address range, whereby the operating system has permission to page out one of the driver resource and the application resource from the system memory corresponding to the committed kernel virtual address range.

27. The method as recited in claim 19, further comprising:

allocating and committing an application private virtual address range for a surface, the surface not being directly accessible by the application via the main processing unit.

28. The method as recited in claim 27, further comprising:

allocating memory in the local video memory for containing the surface; and

causing the surface to be copied from memory corresponding to the committed application private virtual address range to the memory allocated in the local video memory.

29. The method as recited in claim 28, further comprising:

causing the surface to be copied from the memory allocated in the local video memory to memory corresponding to the committed application private virtual address range; and

freeing the memory allocated in the local video memory.

30. The method as recited in claim 27, further comprising:

locking the committed application private virtual address range, whereby the operating system does not have permission to page out the surface from the system memory corresponding to the application private virtual address range;

allocating an address range in the graphics processing unit aperture for redirection to the surface; and

causing the address range in the graphics processing unit aperture to be mapped to the committed application private virtual address range.

31. The method as recited in claim 30, further comprising:

unmapping and freeing the allocated graphics processing unit aperture address range; and

unlocking the committed application private virtual address range, whereby the operating system has permission to page out the surface from the system memory corresponding to the committed application private virtual address range.

32. The method as recited in claim 19, further comprising:

allocating and committing an application private virtual address range for a surface, the surface being directly accessible by the application via the main processing unit.

33. The method as recited in claim 32, further comprising:

allocating memory in the local video memory for containing the surface;

causing the surface to be copied from the memory corresponding to the committed application private virtual address range to the memory allocated in the local video memory; and

mapping the committed application private virtual address range to the memory allocated in the local video memory.

34. The method as recited in claim 33, further comprising:

causing the surface to be copied from the memory allocated in the local video memory to memory corresponding to the committed application private virtual address range;

freeing the allocated memory in the local video memory; and

remapping the committed application private virtual address range to the memory corresponding to the committed application private virtual address range.

35. The method as recited in claim 32, further comprising:

locking the committed application private virtual address range, whereby the operating system does not have permission to page out the surface from the system memory corresponding to the application private virtual address range;

allocating an address range in the graphics processing unit aperture for redirection to the surface; and

causing the address range in the graphics processing unit aperture to be mapped to the committed application private virtual address range.

36. The method as recited in claim 35, further comprising:

unmapping and freeing the allocated graphics processing unit aperture address range; and

unlocking the committed application private virtual address range, whereby the operating system has permission to page out the surface from the system memory corresponding to the committed application private virtual address range.

37. The method as recited in claim 19, further comprising:

determining whether a set of graphics data to be stored in a memory is larger than a predefined size; and

if the set of graphics data is larger than the predefined size, allocating memory that is a multiple of a page size of the operating system and storing only the set of graphics data in the allocated memory; and

if the set of graphics data is not larger than the predefined size, allocating memory that is a multiple of a page size of the operating system and storing the set of graphics data in the allocated memory along with other sets of graphics data.

38. The method as recited in claim 19, further comprising:

querying the graphics processing unit for an indication of a command that was last processed by the graphics processing unit;

receiving, from the graphics processing unit, an indication of a command that was last processed by the graphics processing unit; and

determining which allocation to evict based on the indication of the command that was last processed by the graphics processing unit.

39. The method as recited in claim 19, further comprising:

queuing a marker into a rendering stream communicated to the graphics processing unit;

querying the graphics processing unit for an indication of a marker that was last processed by the graphics processing unit; and

determining which allocation to evict based on the indication of the marker that was last processed by the graphics processing unit.

40. The method as recited in claim 38, wherein the indication of the command that was last processed by the graphics processing unit comprises a monotonic counter that is updated by the graphics processing unit each time a partial command buffer is completed.

41. The method as recited in claim 19, further comprising:

associating a handle with graphics data, the handle being a unique and permanent representation of the graphics data, the graphics data being stored at a physical address;

converting, upon a request including the handle, the handle to the physical address.

42. The method as recited in claim 19, further comprising:

timestamping graphics data with a fence identification; and

marking the graphics data as a candidate for eviction based upon the timestamped fence identification.

43. A method for managing memory in a computer environment having a main processing unit and a plurality of memories, the method comprising:

allocating physical memory in a first one of the plurality of memories;

storing a set of graphics data in the first one of the plurality of memories;

allocating an virtual address range in at least one of the plurality of memories;

mapping the virtual address range to the first one of the plurality of memories storing the graphics data;

storing the graphics data in a second one of the plurality of memories; and

remapping the virtual address range to map to the second one of the plurality of memories.

44. The method as recited in claim 41, further comprising:

determining that a second set of graphics data is a first type of graphics data; and

storing the second set of graphics data in a predefined one of the plurality of memories and not evicting the second set of graphics data from the predefined one of the plurality of memories.

45. The method as recited in claim 44, further comprising:

determining that a second set of graphics data is a second type of graphics data; and

storing the second set of graphics data in one of the plurality of memories and evicting the second set of graphics data from the one of the plurality of memories.

46. The method as recited in claim 41, further comprising:

determining that a second set of graphics data is a first type of graphics data; and

storing the second set of graphics data in a kernel of an operating system of the computer environment.

47. The method as recited in claim 41, further comprising:

distinguishing between a first type of graphics data and a second type of graphics data; and

storing the first type of graphics data in a kernel address space and storing the second type of graphics data in a user address space.

48. A method for video memory management in a computer environment having a main processing unit for executing an operating system and an application, a system memory, and a graphics processing unit having local video memory, the method comprising:

managing the physical memory of the local video memory and at least of portion of the physical memory of the system memory;

allocating virtual memory and maintaining mappings between the allocated virtual memory and the physical memory of the local video memory and the physical memory of the system memory.

49. The method as recited in claim 48, further comprising:

allocating memory for one of a driver resource and an application resource in the local video memory and not evicting the allocated memory from local video memory;

allocating and committing a kernel virtual address range for one of the driver resource and the application resource; and

mapping the kernel virtual address range to the memory allocated in local video memory.

50. The method as recited in claim 48, further comprising:

allocating and committing a kernel virtual address range for one of a driver resource and an application resource.

51. The method as recited in claim 50, further comprising:

allocating memory in the local video memory for containing of one of the driver resource and the application resource;

causing one of the driver resource and the application resource to be copied from memory corresponding to the committed kernel virtual address range to the memory allocated in the local video memory; and

mapping the kernel virtual address range to the memory allocated in the local video memory.

52. The method as recited in claim 51, further comprising:

causing one of the driver resource and the application resource to be copied from the memory allocated in the local video memory to memory corresponding to the committed kernel virtual address range;

freeing the memory allocated in the local video memory; and

freeing the mapped kernel virtual address range.

53. The method as recited in claim 48, further comprising:

allocating and committing an application private virtual address range for a surface, the surface not being directly accessible by the application via the main processing unit.

54. The method as recited in claim 53, further comprising:

allocating memory in the local video memory for containing the surface; and

causing the surface to be copied from memory corresponding to the committed application private virtual address range to the memory allocated in the local video memory.

55. The method as recited in claim 54, further comprising:

causing the surface to be copied from the memory allocated in the local video memory to memory corresponding to the committed application private virtual address range; and

freeing the memory allocated in the local video memory.

56. The method as recited in claim 48, further comprising:

allocating and committing an application private virtual address range for a surface, the surface being directly accessible by the application via the main processing unit.

57. The method as recited in claim 56, further comprising:

allocating memory in the local video memory for containing the surface;

causing the surface to be copied from the memory corresponding to the committed application private virtual address range to the memory allocated in the local video memory; and

mapping the committed application private virtual address range to the memory allocated in the local video memory.

58. The method as recited in claim 57, further comprising:

causing the surface to be copied from the memory allocated in the local video memory to memory corresponding to the committed application private virtual address range;

freeing the allocated memory in the local video memory; and

remapping the committed application private virtual address range to the memory corresponding to the committed application private virtual address range.

59. A method for video memory management in a computer environment having a main processing unit for executing an operating system and an application, a system memory, and a graphics processing unit having an aperture that maps between a portion of system memory and the graphics processing unit, the method comprising:

managing the memory mappings between a portion of system memory and the graphics processing unit, such that video data in the system memory is accessible to the graphics processing unit via the aperture; and

allocating virtual memory and maintaining mappings between the allocated virtual memory and the physical memory of the local video memory, the physical memory of the system memory, and the physical memory of the system memory accessible via the aperture.

60. The method as recited in claim 59, further comprising:

allocating and committing a kernel virtual address range for one of a driver resource and an application resource;

locking the committed kernel virtual address range, whereby the operating system does not have permission to page out one of the driver resource and the application resource from the system memory corresponding to the committed kernel virtual address range;

allocating an address range in the graphics processing unit aperture for redirection to one of the driver resource and the application resource; and

causing the allocated address range in the graphics processing unit aperture to be mapped to the committed kernel virtual address range.

61. The method as recited in claim 59, further comprising:

allocating and committing a kernel virtual address range for one of a driver resource and an application resource.

62. The method as recited in claim 61, further comprising:

locking the committed kernel virtual address range, whereby the operating system does not have permission to page out one of the driver resource and the application resource from the system memory corresponding to the committed kernel virtual address range;

allocating an address range in the graphics processing unit aperture for redirection to one of the driver resource and the application resource; and

causing the address range in the graphics processing unit aperture to be mapped to the committed kernel virtual address range.

63. The method as recited in claim 62, further comprising:

unmapping and freeing the address range allocated in the graphics processing unit aperture; and

unlocking the committed kernel virtual address range, whereby the operating system has permission to page out one of the driver resource and the application resource from the system memory corresponding to the committed kernel virtual address range.

64. The method as recited in claim 59, further comprising:

allocating and committing an application private virtual address range for a surface, the surface not being directly accessible by the application via the main processing unit.

65. The method as recited in claim 64, further comprising:

locking the committed application private virtual address range, whereby the operating system does not have permission to page out the surface from the system memory corresponding to the application private virtual address range;

allocating an address range in the graphics processing unit aperture for redirection to the surface; and

causing the address range in the graphics processing unit aperture to be mapped to the committed application private virtual address range.

66. The method as recited in claim 65, further comprising:

unmapping and freeing the allocated graphics processing unit aperture address range; and

unlocking the committed application private virtual address range, whereby the operating system has permission to page out the surface from the system memory corresponding to the committed application private virtual address range.

67. The method as recited in claim 59, further comprising:

allocating and committing an application private virtual address range for a surface, the surface being directly accessible by the application via the main processing unit.

68. The method as recited in claim 67, further comprising:

locking the committed application private virtual address range, whereby the operating system does not have permission to page out the surface from the system memory corresponding to the application private virtual address range;

allocating an address range in the graphics processing unit aperture for redirection to the surface; and

causing the address range in the graphics processing unit aperture to be mapped to the committed application private virtual address range.

69. The method as recited in claim 68, further comprising:

unmapping and freeing the allocated graphics processing unit aperture address range; and

unlocking the committed application private virtual address range, whereby the operating system has permission to page out the surface from the system memory corresponding to the committed application private virtual address range.